

BDM PROJECT

Team Members

Parul Ghai

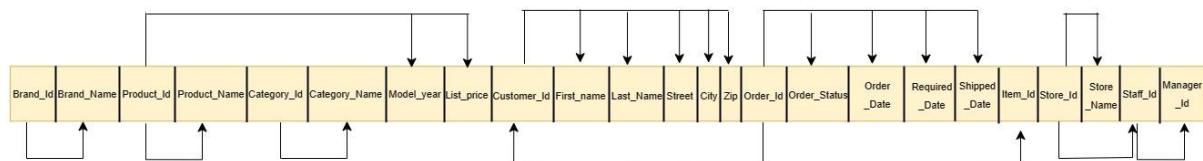
Prateek Gupta

Dataset

The Bike Store SQL dataset project involves designing and implementing a relational database to manage data for a fictional bike store. The dataset comprises eight interconnected tables: customers, orders, order_items, staffs, stores, categories, products, and brands. This project aims to provide a comprehensive solution for tracking customer interactions, sales transactions, inventory management, and employee activities within the bike store.

Data Normalization

Raw data before normalization:



- The data is redundant and has created the following anomalies:
 - **Creation Anomalies:**
 - If a new customer is added, and the order Id is not already present in the respective tables, then there would be a creation anomaly.
 - **Updation Anomalies:**
 - If the name of a store is updated, then all the records that reference that store name would need to be updated to avoid inconsistency.

- **Deletion Anomalies:**

- If the list price of the product is deleted, then all the records related to the product would be deleted, which would result in losing crucial information on other columns.

Normalized data structure:

<u>Brand_Id</u>	Brand_Name
-----------------	------------

<u>Category_Id</u>	Category_Name
--------------------	---------------

<u>Product_Id</u>	Product_Name	Brand_Id	Category_Id	Model_year	List_price
-------------------	--------------	----------	-------------	------------	------------

<u>Customer_Id</u>	First_Name	Last_Name	Street	City	State	Zip_Code
--------------------	------------	-----------	--------	------	-------	----------

<u>Order_Id</u>	Customer_Id	Order_Status	Order_date	Required_Date	Shipped_Date
-----------------	-------------	--------------	------------	---------------	--------------

<u>Order_Id</u>	<u>Item_Id</u>	List_price	Discount	Product_Id
-----------------	----------------	------------	----------	------------

<u>Store_Id</u>	Store_Name
-----------------	------------

<u>Store_Id</u>	<u>Product_Id</u>	Quantity
-----------------	-------------------	----------

<u>Staff_Id</u>	Manager_Id	<u>Store_Id</u>	First_Name	Last_Name
-----------------	------------	-----------------	------------	-----------

- **Customers Table:**

- Customer_ID (Primary Key)
- First_Name
- Last_Name

- Street
- City
- State
- Zip

- **2. Orders Table:**

- Order_ID (Primary Key)
- Customer_ID (Foreign Key)
- Order_Date
- Order_Status
- Required_Date
- Shipped_Date

- **Order_Items Table:**

- Order_ID (Foreign Key)
- Item_ID (Foreign Key)
- List_price
- Dicount
- Product_Id

- **Staffs Table:**

- Staff_ID (Primary Key)
- Store_Id (Primary Key)
- Manager_Id
- First_Name
- Last_Name

- **Stores Table:**

- Store_ID (Primary Key)
- Product_Id(Primary Key)
- Quantity

- **Categories Table:**

- Category_ID (Primary Key)
- Category_Name

- **Products Table:**

- Product_ID (Primary Key)
- Product_Name
- Brand_Id

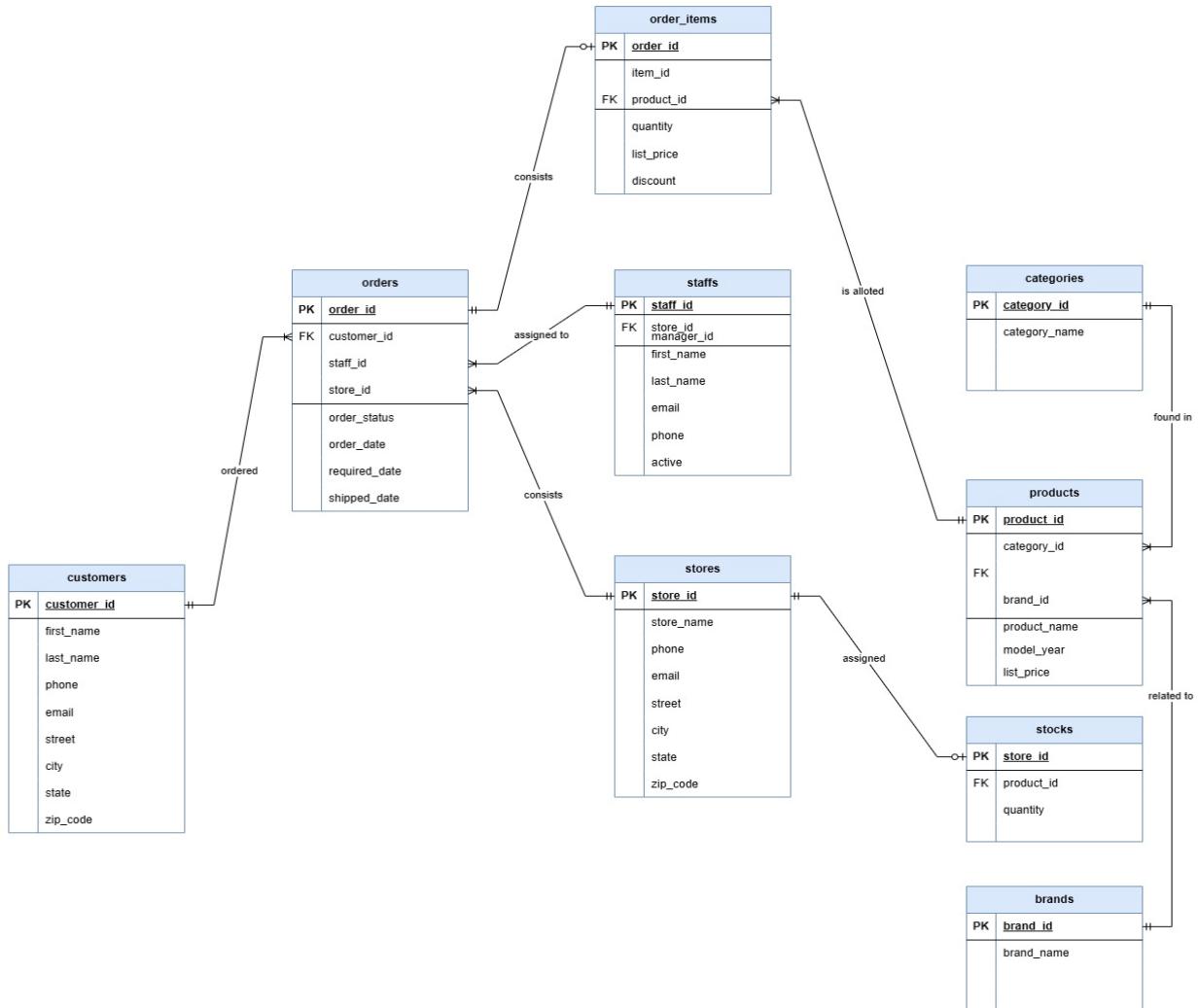
- Category_ID (Foreign Key)
 - Model_Year
 - List_Price
- **Brands Table:**
 - Brand_ID (Primary Key)
 - Brand_Name

Entity-Relation Diagram

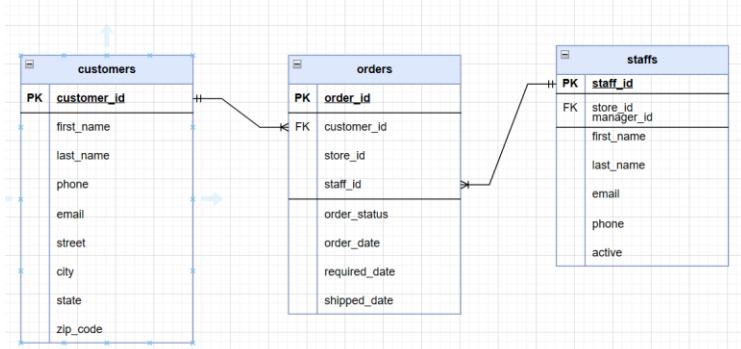
Business Rules:

- The Customers table is linked to the Orders table via the Customer_ID field, establishing a one-to-many relationship, as a customer can place multiple orders.
- The Orders table is connected to the Order_Items table through the Order_ID, allowing a detailed breakdown of items within each order.
- The Order_Items table references both the Orders and Products tables to associate each item with a specific order and product.
- The Staffs and Stores tables provide additional context, linking staff members to specific store locations through order_id from orders table.

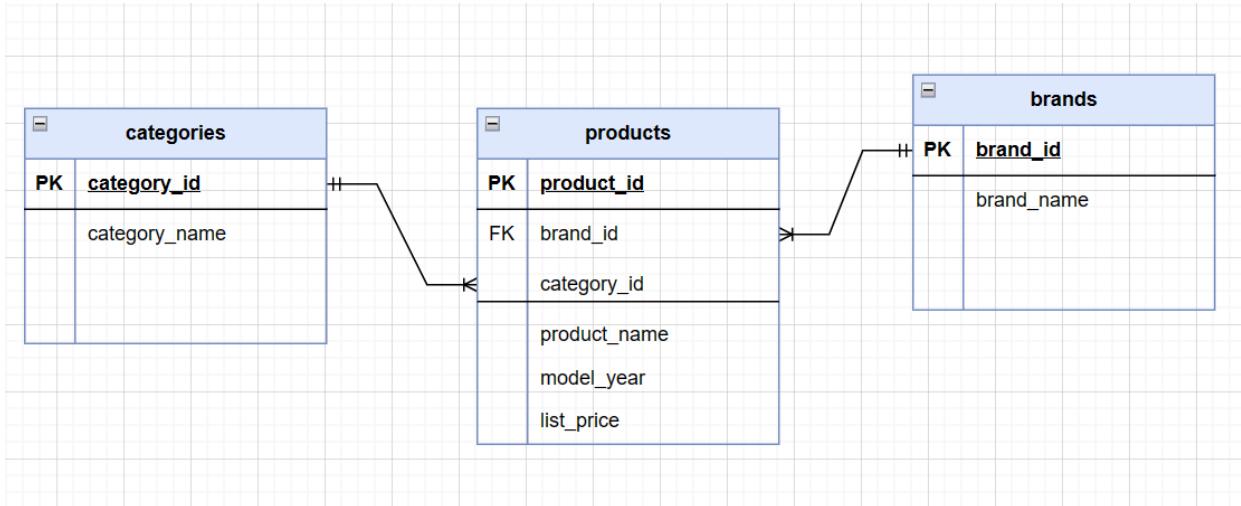
ER Diagram



Customer and Staff have many to many relationship which is bridged by the Orders table.



Categories and Brands have many to many relationship which is bridged by the Product table.



Data Cleaning

Modify column:

```
ALTER TABLE your_table
ALTER COLUMN column_name new_data_type;
```

Renamed all the columns using query:

```
ALTER TABLE your_table
RENAME COLUMN old_name TO new_name;
```

Handle missing data-

Replace missing values:

```
SELECT * FROM `customers` WHERE 1
UPDATE customers
SET phone = (999) 888-7766
WHERE phone IS NULL;
```

Personal MAMP localhost / localhost / bdm / cus MySQL - MySQL 5.7 Reference

localhost/phpMyAdmin/index.php?route=/table/sql&db=bdm&table=customers

Import favorites Amazon.co.uk - On... Gmail YouTube Maps Prime Video India Watch Half Love Ha...

phpMyAdmin

Recent Favorites

- New
- bdm
 - New
 - customers
 - orders
 - order_items
- csv_db 5
- information_schema
- mysql
- performance_schema
- sys

Browse Structure SQL Search Insert Export Import Privileges Operations Triggers

Run SQL query/queries on table bdm.customers:

```

1 SELECT * FROM `customers` WHERE 1
2 UPDATE `customers`
3 SET phone = '(999) 888-7766'
4 WHERE phone IS NULL;
    
```

COL 1
COL 2
COL 3
COL 4
COL 5
COL 6
COL 7
COL 8
COL 9

SELECT*|SELECT|INSERT|UPDATE|DELETE|Clear|Format|Get auto-saved query|Bind parameters

Console Bookmarks Options History Clear

```

Expand Recovery Edit Explain Profiling Bookmark Database : bdm Queried time : 18:43:38
>SELECT * FROM `customers`
>SELECT * FROM `order_items`
>SELECT * FROM `orders`
    
```

41°F Mostly cloudy

Personal MAMP localhost / localhost / bdm / cus MySQL - MySQL 5.7 Reference

localhost/phpMyAdmin/index.php?route=/sql&db=bdm&table=customers&sql_query=SELECT+*+FROM+`customers`+ORDER+BY+`customer_id`+ASC

Import favorites Amazon.co.uk - On... Gmail YouTube Maps Prime Video India Watch Half Love Ha...

phpMyAdmin

Recent Favorites

- New
- bdm
 - New
 - customers
 - orders
 - order_items
- csv_db 5
- information_schema
- mysql
- performance_schema
- sys

Browse Structure SQL Search Insert Export Import Privileges Operations Triggers

Showing rows 0 - 99 (1446 total. Query took 0.0028 seconds) [COL 4: PHONE... - (569) 810-6070...]

SELECT * FROM `customers` ORDER BY `customers`.`COL 4` DESC

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

customer_id	first_name	last_name	phone	email	street	city	state	zip_code
67	Tommie	Mellon	(916) 802-2952	tommie.mellon@gmail.com	8 West Old York St.	Sacramento	CA	95820
959	Christel	Barber	(916) 679-3547	christel.barber@hotmail.com	320 Brianwood Lane	Sacramento	CA	95820
915	Beryl	Bennett	(916) 422-6956	beryl.bennett@aol.com	9563 Edgement St.	Sacramento	CA	95820
5	Charlotte	Rice	(916) 381-6003	charlotte.rice@msn.com	107 River Dr.	Sacramento	CA	95820
231	Jenée	Rasmussen	(916) 219-1774	jenee.rasmussen@hotmail.com	19 George Lane	Sacramento	CA	95820
1052	Zora	Ford	(916) 192-3195	zora.ford@gmail.com	7416 La Sierra Dr.	Sacramento	CA	95820
675	Kermit	Bowman	(915) 996-5952	kermit.bowman@gmail.com	3 Airport Street	El Paso	TX	79930
883	Gilberte	Duke	(915) 903-7860	gilberte.duke@gmail.com	12 Birchwood Dr.	El Paso	TX	79930
584	Lynne	Anderson	(915) 780-6900	lynne.anderson@msn.com	325 Inverness St	El Paso	TX	79930
1024	Coleman	Boyd	(915) 607-6336	coleman.boyd@yahoo.com	9740 Bay Meadows Drive	El Paso	TX	79930
576	Shirley	Stanley	(915) 437-6113	shirley.stanley@gmail.com	73 White Avenue	El Paso	TX	79930
250	Ivonne	Yang	(915) 181-1950	ivonne.yang@gmail.com	7359 North Lake View St.	El Paso	TX	79930

Console Bookmarks Options History Clear

```

>SELECT * FROM `customers`
=UPDATE customers SET phone = 999 WHERE phone = NULL;
=UPDATE customers SET phone = 999 WHERE phone IS NULL;
>SELECT * FROM `customers` ORDER BY `customers`.`COL 4` ASC
    
```

40°

customers

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	customer_first_name	last_name	phone		email	street	city	state	zip_code												
2	1	Debra	Burks	(999) 888-7766	debra.burk	9273 Thor Orchard Pt	NY		14127												
3	2	Kasha	Todd	(999) 888-7766	kasha.todk	910 Vine S	Campbell	CA	95008												
4	3	Tameka	Fisher	(999) 888-7766	tameka.fis	769C Horn Redondo E	CA		90278												
5	4	Daryl	Spence	(999) 888-7766	dary.spen	988 Pearl I	Uniondale	NY	11553												
6	5	Charlotte	Rice	(916) 381-6003	charolette.107	River I	Sacramento	CA	95820												
7	6	Lynsey	Bean	(999) 888-7766	lynsey.be	769 West I	Fairport	NY	14450												
8	7	Latasha	Hays	(716) 986-3359	latasha.ha	7014 Main Buffalo	NY		14215												
9	8	Jacqueline	Duncan	(999) 888-7766	jacqueline.c	15 Brown J	Jackson H	NY	11372												
10	9	Genoveva	Baldwin	(999) 888-7766	genoveva.	8550 Spru Port Wash	NY		11050												
11	10	Pamelia	Newman	(999) 888-7766	pamelia.n	476 Chest Monroe	NY		10950												
12	11	Deshawn	Mendoza	(999) 888-7766	deshawn.r	8790 Cobt Monroe	NY		10952												
13	12	Robby	Syles	(516) 583-7761	robby.yke	486 Rock I	Hempstea	NY	11550												
14	13	Lashawn	Ortiz	(999) 888-7766	lashawn.o	27 Washin Longview	TX		75604												
15	14	Garry	Espinosa	(999) 888-7766	garry.espi	7858 Rock Forney	TX		75126												
16	15	Linnie	Branch	(999) 888-7766	linnie.bran	314 South Plattsburg	NY		12901												
17	16	Emmitt	Sanchez	(212) 945-8823	emmitt.sa	461 Squaw New York	NY		10002												
18	17	Caren	Stephens	(999) 888-7766	caren.step	914 Brook Scarsdale	NY		10583												
19	18	Georgetta	Hardin	(999) 888-7766	georgetta.	474 Chapp Canandaig	NY		14424												
20	19	Lizzette	Stein	(999) 888-7766	lizzette.ste	19 Green I	Orchard Pt	NY	14127												
21	20	Alleta	Shepard	(999) 888-7766	alleta.shep	684 Howa Sugar Lane	TX		77478												
22	21	Tobie	Little	(999) 888-7766	tobie.little	10 Silver St Victoria	TX		77904												
23	22	Adelle	Larsen	(999) 888-7766	adelle.lars	683 West I East North	NY		11731												
24	23	Kaylee	English	(999) 888-7766	kaylene.eng	8786 Fullo Holls	NY		11423												
25	24	Corene	Wall	(999) 888-7766	corene.wa	9601 Ocea Atwater	CA		95301												
26	25	Regenia	Vaughan	(999) 888-7766	regenia.va	44 Stonyb Mahopac	NY		10541												
27	26	Theo	Reese	(562) 215-2907	theo.rees	8755 W. V Long Beach	NY		11561												

Delete rows with missing values:

DELETE FROM orders

WHERE shipped_date IS NULL;

orders

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
922	921	628	4	5/31/2017	6/2/2017	6/2/2017		2	6							
923	922	656	4	5/31/2017	6/1/2017	6/3/2017		2	6							
924	923	769	4	5/31/2017	6/3/2017	6/3/2017		2	7							
925	924	68	3	6/2/2017	6/2/2017	6/3/2017		3	8							
926	925	1316	4	6/2/2017	6/2/2017	6/3/2017		2	7							
927	926	233	4	6/3/2017	6/5/2017	6/6/2017		1	2							
928	927	1438	4	6/3/2017	6/5/2017	6/6/2017		1	2							
929	928	957	4	6/5/2017	6/6/2017	6/6/2017		1	3							
930	929	472	4	6/5/2017	6/6/2017	6/8/2017		2	6							
931	930	1224	4	6/5/2017	6/6/2017	6/6/2017		3	9							
932	931	631	4	6/7/2017	6/8/2017	6/8/2017		2	6							
933	932	804	4	6/7/2017	6/8/2017	6/9/2017		2	6							
934	933	909	4	6/7/2017	6/10/2017	6/8/2017		2	6							
935	934	93	4	6/9/2017	6/10/2017	6/12/2017		2	7							
936	935	43	3	6/10/2017	6/10/2017	NULL		3	8							
937	936	265	4	6/10/2017	6/13/2017	6/12/2017		2	6							
938	937	73	4	6/11/2017	6/14/2017	6/13/2017		2	7							
939	938	129	4	6/11/2017	6/12/2017	6/12/2017		2	6							
940	939	1407	4	6/11/2017	6/12/2017	6/14/2017		2	6							
941	940	585	4	6/12/2017	6/14/2017	6/13/2017		1	2							
942	941	736	4	6/12/2017	6/14/2017	6/14/2017		1	2							
943	942	537	4	6/13/2017	6/16/2017	6/15/2017		2	6							
944	943	1025	4	6/13/2017	6/15/2017	6/15/2017		2	6							
945	944	138	4	6/14/2017	6/15/2017	6/17/2017		1	3							
946	945	167	4	6/14/2017	6/15/2017	6/16/2017		2	7							
947	946	525	4	6/14/2017	6/16/2017	6/15/2017		2	6							
948	947	278	4	6/15/2017	6/18/2017	6/17/2017		1	3							

Personal MAMP localhost / localhost / bdm / ord MySQL - MySQL 5.7 Reference

localhost/phpMyAdmin/index.php?route=/sql&server=1&db=bdm&table=orders&pos=0

Import favorites Amazon.co.uk - Onl... Gmail YouTube Maps Prime Video India Watch Half Love Ha...

phpMyAdmin

Recent Favorites

New bdm New customers orders order_items csv_db 5 information_schema mysql performance_schema sys

Browse Structure SQL Search Insert Export Import Privileges Operations Triggers

Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available.

Showing rows 0 - 499 (1616 total, Query took 0.0012 seconds.)

SELECT * FROM `orders`

Profiling | Edit inline | Edit | Explain SQL | Create PHP code | Refresh

Number of rows: 500 Filter rows: Search this table

COL 1	COL 2	COL 3	COL 4	COL 5	COL 6	COL 7	COL 8
order_id	customer_id	order_status	order_date	required_date	shipped_date	store_id	staff_id
1	259	4	2016-01-01	2016-01-03	2016-01-03	1	2
2	1212	4	2016-01-01	2016-01-04	2016-01-03	2	6
3	523	4	2016-01-02	2016-01-05	2016-01-03	2	7
4	175	4	2016-01-03	2016-01-04	2016-01-05	1	3
5	1324	4	2016-01-03	2016-01-06	2016-01-06	2	6
6	94	4	2016-01-04	2016-01-07	2016-01-05	2	6
7	324	4	2016-01-04	2016-01-07	2016-01-05	2	6
8	1204	4	2016-01-04	2016-01-05	2016-01-05	2	7
9	60	4	2016-01-05	2016-01-08	2016-01-08	1	2
10	442	4	2016-01-05	2016-01-06	2016-01-06	2	6
11	1396	4	2016-01-05	2016-01-08	2016-01-07	2	7

Console

Press Ctrl+Enter to execute query

```
>SELECT * FROM `customers`
>SELECT * FROM `customers`
>SELECT * FROM `order items`
```

Bookmarks Options History Clear

Mostly cloudy 7:32 PM 12/20/2023

AutoSave off orders

File Home Insert Page Layout Formulas Data Review View Automate Help

Font Alignment Number Conditional Formatting Styles Cells Editing

F1011 : x fx NULL

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1000	999	590	4	7/12/2017	7/13/2017	7/14/2017	2	6						
1001	1000	645	4	7/12/2017	7/14/2017	7/15/2017	2	7						
1002	1001	354	4	7/13/2017	7/16/2017	7/15/2017	2	6						
1003	1002	1418	4	7/14/2017	7/16/2017	7/16/2017	1	3						
1004	1003	503	4	7/14/2017	7/16/2017	7/16/2017	2	7						
1005	1004	1389	4	7/14/2017	7/17/2017	7/16/2017	2	6						
1006	1005	266	4	7/15/2017	7/17/2017	7/16/2017	2	7						
1007	1006	1070	4	7/15/2017	7/18/2017	7/17/2017	2	7						
1008	1007	1115	4	7/16/2017	7/17/2017	7/17/2017	2	6						
1009	1008	1345	4	7/16/2017	7/18/2017	7/18/2017	2	6						
1010	1009	1064	4	7/16/2017	7/19/2017	7/19/2017	3	8						
1011	1010	8	3	7/18/2017	7/18/2017	NULL	2	7						
1012	1011	495	4	7/18/2017	7/21/2017	7/21/2017	1	3						
1013	1012	620	4	7/18/2017	7/20/2017	7/20/2017	2	7						
1014	1013	174	4	7/19/2017	7/22/2017	7/21/2017	2	6						
1015	1014	748	4	7/19/2017	7/21/2017	7/21/2017	2	6						
1016	1015	799	4	7/19/2017	7/22/2017	7/21/2017	2	7						
1017	1016	1417	4	7/19/2017	7/20/2017	7/21/2017	3	9						
1018	1017	168	4	7/20/2017	7/21/2017	7/23/2017	2	6						
1019	1018	290	4	7/22/2017	7/23/2017	7/23/2017	2	6						
1020	1019	408	4	7/22/2017	7/23/2017	7/24/2017	2	7						
1021	1020	16	3	7/23/2017	7/23/2017	NULL	2	6						
1022	1021	125	4	7/23/2017	7/24/2017	7/26/2017	2	6						
1023	1022	353	4	7/23/2017	7/26/2017	7/24/2017	2	7						
1024	1023	356	4	7/23/2017	7/26/2017	7/25/2017	2	6						
1025	1024	837	4	7/23/2017	7/25/2017	7/26/2017	2	7						
1026	1025	1304	4	7/23/2017	7/26/2017	7/26/2017	2	6						

orders

Select destination and press ENTER or choose Paste

Mostly cloudy 7:04 PM 12/20/2023

The screenshot displays two separate sessions in the phpMyAdmin interface.

Session 1 (Top):

- Panel: Run SQL query/queries on table bdm.orders
- Query:

```

1 SELECT * FROM `orders` WHERE 1
2 DELETE FROM orders
3 WHERE shipped_date IS NULL;
        
```

- Buttons: SELECT*, INSERT, UPDATE, DELETE, Clear, Format, Get auto-saved query
- Console:

```

>SELECT * FROM `customers`
>SELECT * FROM `customers`
>SELECT * FROM `orders`
        
```

Session 2 (Bottom):

- Panel: Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available.
- Message: Showing rows 0 - 499 (1618 total, Query took 0.0000 seconds.)
- Query:

```
SELECT * FROM `orders`
```

- Buttons: Profiling, Edit inline, Explain SQL, Create PHP code, Refresh
- Table View (grid):

COL_1	COL_2	COL_3	COL_4	COL_5	COL_6	COL_7	COL_8
1	259	4	2016-01-01	2016-01-03	2016-01-03	1	2
2	1212	4	2016-01-01	2016-01-04	2016-01-03	2	6
3	523	4	2016-01-02	2016-01-05	2016-01-03	2	7
4	175	4	2016-01-03	2016-01-04	2016-01-05	1	3
5	1324	4	2016-01-03	2016-01-06	2016-01-06	2	6
6	94	4	2016-01-04	2016-01-07	2016-01-05	2	6
7	324	4	2016-01-04	2016-01-07	2016-01-05	2	6
8	1204	4	2016-01-04	2016-01-05	2016-01-05	2	7
9	60	4	2016-01-05	2016-01-08	2016-01-08	1	2
10	442	4	2016-01-05	2016-01-06	2016-01-06	2	6
11	1396	4	2016-01-05	2016-01-08	2016-01-07	2	7
- Console:

```

Press Ctrl+Enter to execute query
>SELECT * FROM `customers`
>SELECT * FROM `customers`
>SELECT * FROM `order_items`
        
```

Deal with outliers:

Identify outliers:

`SELECT * FROM your_table`

`WHERE column_name > upper_bound OR column_name < lower_bound;`

CREATE SQL CODE AND IMPORT DATASET:

Use Sql workshop> utilities> data workshop> Load function on apex oracle to upload csv files

Then used insert statements to insert data from uploaded tables into created tables in mySQL

1) BRANDS

```
1 CREATE TABLE brands (
2   brand_id INT PRIMARY KEY,
3   brand_name VARCHAR(50) NOT NULL
4 );
```

A	B	C
1	brand_id	brand_name
2	1	Electra
3	2	Haro
4	3	Heller
5	4	Pure Cycles
6	5	Ritchey
7	6	Strider
8	7	Sun Bicycles
9	8	Surly
10	9	Trek

Table has brand_id as primary key and columns giving information about different brands.

2) Categories:

```
1 CREATE TABLE categories (
2   category_id INT PRIMARY KEY,
3   category_name VARCHAR(50) NOT NULL
4 );
```

A	B	C
1	category_id	category_name
2	1	Children Bicycles
3	2	Comfort Bicycles
4	3	Cruisers Bicycles
5	4	Cyclocross Bicycles
6	5	Electric Bikes
7	6	Mountain Bikes
8	7	Road Bikes
9		

Table has category_id as primary key and columns giving information about different categories.

3) Customers

```
1 CREATE TABLE customers (
2     customer_id INT PRIMARY KEY,
3     first_name VARCHAR(50) NOT NULL,
4     last_name VARCHAR(50),
5     phone VARCHAR(20),
6     email VARCHAR(100),
7     street VARCHAR(100),
8     city VARCHAR(50),
9     state VARCHAR(2),
10    zip_code VARCHAR(10)
11 );
12 
```

A	B	C	D	E	F	G	H	I	J
customer	first_name	last_name	phone	email	street	city	state	zip_code	
1	Debra	Burks	NULL	debra.burks@9273 Thorne	Orchard Park	NY	14127		
2	Kasha	Todd	NULL	kasha.todd@910 Vine Stre	Campbell	CA	95008		
3	Tameka	Fisher	NULL	tameka.fisher769C Honey	(Redondo Beach	CA	90278		
4	Daryl	Spence	NULL	daryl.spence@988 Pearl	Lan Uniondale	NY	11553		
5	Charolette	Rice	(916) 381-600	charolette.ric	107 River Dr.	Sacramento	CA	95820	
6	Lyndsey	Bean	NULL	lyndsey.bean@769 West	Ros Fairport	NY	14450		
7	Latasha	Hays	(716) 986-335	latasha.hays@7014	Manor	Buffalo	NY	14215	
8	Jacqueline	Duncan	NULL	jacqueline.dun	15 Brown St.	Jackson Height	NY	11372	
9	Genoveva	Baldwin	NULL	genoveva.bal	8550 Spruce	(Port Washingt	NY	11050	
10	Pamelia	Newman	NULL	pamelia.newr	476 Chestnut Monroe	NY	10950		
11	Deshawn	Mendoza	NULL	deshawn.mer	8790 Cobble	Monsey	NY	10952	
12	Robby	Sykes	(516) 583-776	robby.sykes@486 Rock	May	Hempstead	NY	11550	
13	Lashawn	Ortiz	NULL	lashawn.ortiz@27	Washington	Longview	TX	75604	
14	Garry	Espinosa	NULL	garry.espinoz	7858 Rockaw	Forney	TX	75126	
15	Linnie	Branch	NULL	linnie.branchi	314 South Co	Plattsburgh	NY	12901	
16	Emmit	Sanchez	(212) 945-882	emmitt.sanch	461 Squaw Cr	New York	NY	10002	
17	Caren	Stephens	NULL	caren.stepher	914 Brook St.	Scarsdale	NY	10583	
18	Georgetta	Hardin	NULL	georgetta.har	474 Chapel D	Canandaigua	NY	14424	
19	Lizzette	Stein	NULL	lizzette.stein@19	Green Hill	Orchard Park	NY	14127	
20	Aleta	Shepard	NULL	aleta.shepard	684 Howard	Sugar Land	TX	77478	
21									

Table has customer_id as primary key and columns giving information about contact and address of customers.

4) Order_items

```
1 CREATE TABLE order_items (
2     order_id INT,
3     item_id INT,
4     product_id INT,
5     quantity INT,
6     list_price DECIMAL(10, 2),
7     discount DECIMAL(4, 2),
8     PRIMARY KEY (order_id, item_id),
9     FOREIGN KEY (product_id) REFERENCES products(product_id)
10 );
11 
```

	A	B	C	D	E	F	G
1	order_id	item_id	product_i	quantity	list_price	discount	
2	1	1	20	1	599.99	0.2	
3	1	2	8	2	1799.99	0.07	
4	1	3	10	2	1549	0.05	
5	1	4	16	2	599.99	0.05	
6	1	5	4	1	2899.99	0.2	
7	2	1	20	1	599.99	0.07	
8	2	2	16	2	599.99	0.05	
9	3	1	3	1	999.99	0.05	
10	3	2	20	1	599.99	0.05	
11	4	1	2	2	749.99	0.1	
12	5	1	10	2	1549	0.05	
13	5	2	17	1	429	0.07	
14	5	3	26	1	599.99	0.07	
15	6	1	18	1	449	0.07	
16	6	2	12	2	549.99	0.05	
17	6	3	20	1	599.99	0.1	
18	6	4	3	2	999.99	0.07	
19	6	5	9	2	2999.99	0.07	
20	7	1	15	1	529.99	0.07	
21	7	2	3	1	999.99	0.1	

Table has primary key(order_id, item_id) and columns giving information about product and prices.

5) Orders

	A	B	C	D	E	F	G	H	I
1	order_id	customer_id	order_status	order_date	required_date	shipped_date	store_id	staff_id	
2	1	259	4	1/1/2016	1/3/2016	1/3/2016	1	2	
3	2	1212	4	1/1/2016	1/4/2016	1/3/2016	2	6	
4	3	523	4	1/2/2016	1/5/2016	1/3/2016	2	7	
5	4	175	4	1/3/2016	1/4/2016	1/5/2016	1	3	
6	5	1324	4	1/3/2016	1/6/2016	1/6/2016	2	6	
7	6	94	4	1/4/2016	1/7/2016	1/5/2016	2	6	
8	7	324	4	1/4/2016	1/7/2016	1/5/2016	2	6	
9	8	1204	4	1/4/2016	1/5/2016	1/5/2016	2	7	
10	9	60	4	1/5/2016	1/6/2016	1/8/2016	1	2	
11	10	442	4	1/5/2016	1/6/2016	1/6/2016	2	6	
12	11	1326	4	1/5/2016	1/8/2016	1/7/2016	2	7	
13	12	91	4	1/6/2016	1/8/2016	1/9/2016	1	2	
14	13	873	4	1/8/2016	1/11/2016	1/11/2016	2	6	
15	14	258	4	1/9/2016	1/11/2016	1/12/2016	1	3	
16	15	450	4	1/9/2016	1/10/2016	1/12/2016	2	7	
17	16	552	4	1/12/2016	1/15/2016	1/15/2016	1	3	
18	17	1175	4	1/12/2016	1/14/2016	1/14/2016	1	3	
19	18	541	4	1/14/2016	1/17/2016	1/15/2016	1	3	
20	19	696	4	1/14/2016	1/17/2016	1/16/2016	1	2	
21	20	923	4	1/14/2016	1/16/2016	1/17/2016	1	2	

```

1 CREATE TABLE orders (
2   order_id INT PRIMARY KEY,
3   customer_id INT,
4   order_status INT,
5   order_date DATE,
6   required_date DATE,
7   shipped_date DATE,
8   store_id INT,
9   staff_id INT,
10  FOREIGN KEY (customer_id) REFERENCES customers(customer_id),
11  FOREIGN KEY (store_id) REFERENCES stores(store_id),
12  FOREIGN KEY (staff_id) REFERENCES staff(staff_id)
13 );

```

Table has order_id as primary key and columns giving information about Orders.

6) Products

```
1 CREATE TABLE products (
2     product_id INT PRIMARY KEY,
3     product_name VARCHAR(255) NOT NULL,
4     brand_id INT,
5     category_id INT,
6     model_year INT,
7     list_price DECIMAL(10, 2),
8     FOREIGN KEY (brand_id) REFERENCES brands(brand_id),
9     FOREIGN KEY (category_id) REFERENCES categories(category_id)
10 );|
```

A	B	C	D	E	F	G
product_id	product_name	brand_id	category_id	model_year	list_price	
1	Trek 820 - 2016	9	6	2016	379.99	
2	2 Ritchey Timberwolf Frame	5	6	2016	749.99	
3	3 Surly Wednesday Frameset	8	6	2016	999.99	
4	4 Trek Fuel EX 8 29 - 2016	9	6	2016	2899.99	
5	5 Heller Shagamaw Frame -	3	6	2016	1320.99	
6	6 Surly Ice Cream Truck Frame	8	6	2016	469.99	
7	7 Trek Slash 8 27.5 - 2016	9	6	2016	3999.99	
8	8 Trek Remedy 29 Carbon F	9	6	2016	1799.99	
9	9 Trek Conduit+ - 2016	9	5	2016	2999.99	
10	10 Surly Straggler - 2016	8	4	2016	1549	
11	11 Surly Straggler 650b - 2016	8	4	2016	1680.99	
12	12 Electra Townie Original 21	1	3	2016	549.99	
13	13 Electra Cruiser 1 (24-Inch)	1	3	2016	269.99	
14	14 Electra Girl's Hawaii 1 (16-	1	3	2016	269.99	
15	15 Electra Moto 1 - 2016	1	3	2016	529.99	
16	16 Electra Townie Original 7C	1	3	2016	599.99	
17	17 Pure Cycles Vine 8-Speed	4	3	2016	429	
18	18 Pure Cycles Western 3-Spi	4	3	2016	449	
19	19 Pure Cycles William 3-Spe	4	3	2016	449	
20	20 Electra Townie Original 7C	1	3	2016	599.99	

Table has product_id as primary key and columns giving information about products.

7) Staffs

```
1 CREATE TABLE staff (
2     staff_id INT PRIMARY KEY,
3     first_name VARCHAR(50) NOT NULL,
4     last_name VARCHAR(50) NOT NULL,
5     email VARCHAR(100) NOT NULL,
6     phone VARCHAR(20) NOT NULL,
7     active INT NOT NULL,
8     store_id INT,
9     manager_id INT,
10    FOREIGN KEY (store_id) REFERENCES stores(store_id),
11    FOREIGN KEY (manager_id) REFERENCES staff(staff_id)
12 );|
```

A	B	C	D	E	F	G	H
staff_id	first_name	last_name	email	phone	active	store_id	manager_id
1	Fabiola	Jackson	fabiola.jackson@(831) 555-5554		1	1	NULL
2	Mireya	Copeland	mireya.copelar@(831) 555-5555		1	1	1
3	Genna	Serrano	genna.serrano@(831) 555-5556		1	1	2
4	Virgie	Wiggins	virgie.wiggins@(831) 555-5557		1	1	2
5	Jannette	David	jannette.david@(516) 379-4444		1	2	1
6	Marcelene	Boyer	marcelene.boyer@(516) 379-4445		1	2	5
7	Venita	Daniel	venita.daniel@(516) 379-4446		1	2	5
8	Kali	Vargas	kali.vargas@bil@(972) 530-5555		1	3	1
9	Layla	Terrell	layla.terrell@b@(972) 530-5556		1	3	7
10	Bernardine	Houston	bernardine.ho@(972) 530-5557		1	3	7

Table has staff_id as primary key and columns giving information about staff members.

8) Stocks

```
1 CREATE TABLE stocks (
2     store_id INT,
3     product_id INT,
4     quantity INT,
5     PRIMARY KEY (store_id, product_id),
6     FOREIGN KEY (store_id) REFERENCES stores(store_id),
7     FOREIGN KEY (product_id) REFERENCES products(product_id)
8 );
```

A	B	C	D
store_id	product_id	quantity	
1	1	1	27
2	1	2	5
3	1	3	6
4	1	4	23
5	1	5	22
6	1	6	0
7	1	7	8
8	1	8	0
9	1	9	11
10	1	10	15
11	1	11	8
12	1	12	16
13	1	13	13
14	1	14	8
15	1	15	3
16	1	16	4
17	1	17	2
18	1	18	16
19	1	19	4
20	1	20	26

Table has primary key(store_id, product_id) and columns giving information about stocks of products.

9) Stores

```
1 CREATE TABLE stores (
2     store_id INT PRIMARY KEY,
3     store_name VARCHAR(100) NOT NULL,
4     phone VARCHAR(20) NOT NULL,
5     email VARCHAR(100) NOT NULL,
6     street VARCHAR(255) NOT NULL,
7     city VARCHAR(50) NOT NULL,
8     state VARCHAR(2) NOT NULL,
9     zip_code VARCHAR(10) NOT NULL
10 );
```

A	B	C	D	E	F	G	H	I
store_id	store_name	phone	email	street	city	state	zip_code	
1	Santa Cruz Bikes	(831) 476-4321	santacruz@bikes.s	3700 Portola Drive	Santa Cruz	CA	95060	
2	Baldwin Bikes	(516) 379-8888	baldwin@bikes.sh	4200 Chestnut Lan	Baldwin	NY	11432	
3	Rowlett Bikes	(972) 530-5555	rowlett@bikes.sho	8000 Fairway Aver	Rowlett	TX	75088	

Table has store_id as primary key and columns giving information about different stores.

Queries:

QUERY 1:

```
select category_name  
From CATEGORIES
```

```
1  select category_name  
2  From CATEGORIES
```

Output:

CATEGORY_NAME
Children Bicycles
Comfort Bicycles
Cruisers Bicycles
Cyclocross Bicycles
Electric Bikes
Mountain Bikes
Road Bikes

Explanation:

- It provides all the categories of bike from category table.

QUERY 2:

```
SELECT DISTINCT Product_Name  
FROM PRODUCTS
```

```
1  SELECT DISTINCT Product_Name  
2  FROM PRODUCTS
```

Output:

PRODUCT_NAME
Surly Big Fat Dummy Frameset - 2018
Trek Procaliber Frameset - 2018
Trek Domane SL 5 Disc Women's - 2018
Surly Pack Rat - 2018
Trek CrossRip+ - 2018
Trek Neko+ - 2018
Trek Powerfly 7 FS - 2018
Trek Crockett 5 Disc - 2018

Explanation:

- It provides Unique product names of bike from products table.

QUERY 3:

```
SELECT count(*) as customers_by_state,state
FROM customers
GROUP BY state;
```

```
1  SELECT count(*) as customers_by_state,state
2  FROM customers
3  GROUP BY state;
```

Output:

CUSTOMERS_BY_STATE	STATE
1019	NY
142	TX
284	CA

3 rows returned in 0.01 seconds [Download](#)

Explanation:

- This counts the total number of customers in each state.

QUERY 4:

```
SELECT ROUND(AVG(discount),2) as avg_discount
FROM ORDER_ITEMS;
```

```
1  SELECT ROUND(AVG(discount),2) as avg_discount
2  FROM ORDER_ITEMS;
```

Output:

Results	Explain	Describe	Saved SQL	History
AVG_DISCOUNT				
.11				
1 rows returned in 0.01 seconds Download				

Explanation:

It is essential to get the information of what average discounts the customers are getting.

The ROUND function in SQL is used to round a numeric value to a specified number of decimal places.

QUERY 5:

```
SELECT SUM(quantity) AS total_items_ordered
FROM ORDER_ITEMS;
```

```
1  SELECT SUM(quantity) AS total_items_ordered
2  FROM ORDER_ITEMS;
```

Output:

Results	Explain	Describe	Saved SQL	History
TOTAL_ITEMS_ORDERED				
7078				
1 rows returned in 0.00 seconds Download				

Explanation:

Use sum function to get total items quantity that is sold.

QUERY 6:

```
SELECT product_id,discount
FROM ORDER_ITEMS
WHERE discount <= 0.2
ORDER BY discount DESC;
```

```
1  SELECT product_id,discount
2  FROM ORDER_ITEMS
3  WHERE discount <= 0.2
4  ORDER BY discount DESC;
```

Output:

Results Explain Describe Saved SQL History

PRODUCT_ID	DISCOUNT
2	.2
4	.2
11	.2
10	.2
8	.2
20	.2
6	.2
2	.2

Explanation:

- This is important because it will indicate what all products have the low discount to buy.

QUERY 7:

```
SELECT ROUND(AVG(list_price * ( 1 - discount)),2) AS avg_SALEPRICE_usd
FROM ORDER_ITEMS
WHERE list_price >=1000;
```

```
1  SELECT ROUND(AVG(list_price * ( 1 - discount)),2) AS avg_SALEPRICE_usd
2  FROM ORDER_ITEMS
3  WHERE list_price >=1000;
```

Output:

Results Explain Describe Saved SQL History

AVG_SALEPRICE_USD
2542.02
1 rows returned in 0.01 seconds Download

Explanation:

- This information is important because it will indicate the value of sales more than 1000.

QUERY 8-

```
SELECT *
FROM CUSTOMERS
WHERE state IN ('CA', 'NY')
```

```
1  SELECT *
2  FROM CUSTOMERS
3  WHERE state IN ('CA', 'NY')
```

Output:

Results Explain Describe Saved SQL History

CUSTOMER_ID	FIRST_NAME	LAST_NAME	PHONE	EMAIL	STREET	CITY	STATE	ZIP_CODE
79	Ashanti	Parks	NULL	ashanti.parks@hotmail.com	846 N. Helen St.	Baldwin	NY	11510
80	Sarai	Mckee	(716) 912-8110	sarai.mckee@msn.com	641 Glenwood Avenue	Buffalo	NY	14215
81	Zina	Bonner	NULL	zina.bonner@hotmail.com	9312 8th Street	San Lorenzo	CA	94580
82	Lizzie	Joyner	NULL	lizzie.joyner@msn.com	8541 Roberts St.	Coachella	CA	92236
83	Tiesha	Daniel	NULL	tiesha.daniel@yahoo.com	6 West Bohemia Lane	Scarsdale	NY	10583
84	Armand	Whitehead	NULL	armand.whitehead@hotmail.com	639 Harvey St.	Lindenhurst	NY	11757
85	Teofila	Fischer	NULL	teofila.fischer@aol.com	1 West Brickyard St.	Huntington Station	NY	11746
86	Lissa	Vargas	NULL	lissa.vargas@yahoo.com	517 Victoria Ave.	Oswego	NY	13126

pg611@scarletmail.rutgers.edu bdmparul en Copyright © 1999, 2025, Oracle and/or its affiliates. Oracle APEX 23.2.1

Explanation: This SQL query selects all columns from the "CUSTOMERS" table where the state is either California or New York, filtering the results based on the specified conditions.

QUERY 9–

```
SELECT * FROM PRODUCTS
WHERE product_name LIKE 'T%'
```

```
1  SELECT *
2  FROM PRODUCTS
3  WHERE product_name LIKE 'T%'
```

Results Explain Describe Saved SQL History

ID	PRODUCT_ID	PRODUCT_NAME	BRAND_ID	CATEGORY_ID	MODEL_YEAR	LIST_PRICE
125	125	Trek Kids' Dual Sport - 2018	9	6	2018	469.99
129	129	Trek X-Caliber 7 - 2018	9	6	2018	919.99
130	130	Trek Stache Carbon Frameset - 2018	9	6	2018	919.99
132	132	Trek Procal AL Frameset - 2018	9	6	2018	1499.99
133	133	Trek Procaliber Frameset - 2018	9	6	2018	1499.99
134	134	Trek Remedy 27.5 C Frameset - 2018	9	6	2018	1499.99
135	135	Trek X-Caliber Frameset - 2018	9	6	2018	1499.99
136	136	Trek Procaliber 6 - 2018	9	6	2018	1799.99

pg611@scarletmail.rutgers.edu bdmparul en Copyright © 1999, 2025, Oracle and/or its affiliates. Oracle APEX 23.2.1

Explanation: This SQL query selects all columns from the "PRODUCTS" table where the product name begins with the letter 'T', using the LIKE operator with a pattern match.

QUERY 10 :

```
SELECT COUNT(*) as sold_amount, staff_id
FROM orders
GROUP BY staff_id
ORDER BY sold_amount DESC
```

```

1  SELECT COUNT(*) AS sold_amount, staff_id
2  FROM orders
3  GROUP BY staff_id
4  ORDER BY sold_amount DESC

```

Output:

Results		Explain	Describe	Saved SQL	History
		SOLD_AMOUNT		STAFF_ID	
	553			6	
	540			7	
	184			3	
	164			2	
	88			8	
	86			9	

6 rows returned in 0.01 seconds [Download](#)

Explanation: This SQL query counts the number of orders handled by each staff member, groups the results by staff ID, and orders the output to display the top staff members based on the count of orders in descending order

QUERY 11 :

```

SELECT product_id,
SUM(quantity) AS total_sold,
ROUND(SUM(list_price * (1 - discount) * quantity),2) AS total_price_w_discount
FROM ORDER_ITEMS
GROUP BY product_id
ORDER BY total_sold DESC;

```

```

1  SELECT product_id,
2  SUM(quantity) AS total_sold,
3  ROUND(SUM(list_price * (1 - discount) * quantity),2) AS total_price_w_discount
4  FROM ORDER_ITEMS
5  GROUP BY product_id
6  ORDER BY total_sold DESC;

```

Output:

Results Explain Describe Saved SQL History

PRODUCT_ID	TOTAL SOLD	TOTAL PRICE W DISCOUNT
6	167	70371.6
13	157	37992.99
16	156	82744.62
7	154	555558.61
23	154	41011.63
12	153	75772.12
11	151	226765.55
25	148	66568.67

pg611@scarletmail.rutgers.edu bdmparul en Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 23.2

Explanation: It Provides the hot product id in the market and its total value for sale.

QUERY 12:

```

SELECT
    ord.product_id,
    SUM(ord.quantity) AS quant,
    ord.list_price,
    prod.product_name
FROM
    order_items ord
INNER JOIN
    products prod ON ord.product_id = prod.product_id
GROUP BY
    ord.product_id, ord.list_price, prod.product_name
ORDER BY
    quant DESC;

```

```

1  SELECT
2      ord.product_id,
3      SUM(ord.quantity) AS quant,
4      ord.list_price,
5      prod.product_name
6  FROM
7      order_items ord
8  INNER JOIN
9      products prod ON ord.product_id = prod.product_id
10 GROUP BY
11      ord.product_id, ord.list_price, prod.product_name
12 ORDER BY
13      quant DESC;

```

Output:

Results Explain Describe Saved SQL History

PRODUCT_ID	QUANT	LIST_PRICE	PRODUCT_NAME
6	167	469.99	Surly Ice Cream Truck Frameset - 2016
13	157	2699.99	Electra Cruiser 1 (24-Inch) - 2016
16	156	599.99	Electra Townie Original 7D EQ - 2016
23	154	299.99	Electra Girl's Hawaii 1 (20-inch) - 2015/2016
7	154	3999.99	Trek Slash 8 27.5 - 2016
12	153	549.99	Electra Townie Original 21D - 2016
11	151	1680.99	Surly Straggler 650b - 2016
25	148	499.99	Electra Townie Original 7D - 2015/2016

👤 pg611@scarletmail.rutgers.edu 📄 bdmparul 🌐 en Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 23.2.1

Explanation:

- The query retrieves product details, including the total quantity sold (quant), list price, and product name, by joining the order_items and products tables, grouping by product ID, list price, and product name, and ordering the results by total quantity in descending order.

QUERY 13:

Select

```
Count(o.order_id) as sold_amount,
o.staff_id, o.store_id, s.first_name, s.last_name
from Orders o
Join staffs s ON o.staff_id = s.staff_id
GROUP BY o.staff_id, o.store_id, s.first_name, s.last_name
```

```
1  SELECT
2    COUNT(o.order_id) as sold_amount,
3    o.staff_id,
4    o.store_id,
5    s.first_name,
6    s.last_name
7  FROM ORDERS o
8  JOIN STAFFS s ON o.staff_id = s.staff_id
9  GROUP BY o.staff_id, o.store_id, s.first_name, s.last_name
10 ORDER BY sold_amount DESC
11  FETCH FIRST 3 ROWS ONLY;
12
```

Output:

Results Explain Describe Saved SQL History

SOLD_AMOUNT	STAFF_ID	STORE_ID	FIRST_NAME	LAST_NAME
553	6	2	Marcelene	Boyer
540	7	2	Venita	Daniel
184	3	1	Genna	Serrano

3 rows returned in 0.02 seconds [Download](#)

Explanation: It uses the COUNT function to calculate the number of orders handled by each staff member, joins the orders and staffs tables based on the staff ID, groups the results by staff ID, store ID, first name, and last name, and then orders the results in descending order of the order count. The FETCH FIRST 3 ROWS ONLY clause limits the output to the top 3 staff members.

QUERY 14:

```
SELECT
    COUNT(quantity) as quantity_count,
    stores.store_name,
    stores.state
FROM STOCKS
INNER JOIN STORES
ON stocks.store_id = stores.store_id
GROUP BY stores.store_id, stores.store_name, stores.state
```

```
1  SELECT
2      COUNT(quantity) as quantity_count,
3      stores.store_name,
4      stores.state
5  FROM STOCKS
6  INNER JOIN STORES
7  ON stocks.store_id = stores.store_id
8  GROUP BY stores.store_id, stores.store_name, stores.state
```

Output:

Results		
QUANTITY_COUNT		
	STORE_NAME	STATE
313	Santa Cruz Bikes	CA
313	Baldwin Bikes	NY
313	Rowlett Bikes	TX

3 rows returned in 0.00 seconds [Download](#)

Explanation: This SQL query counts the quantities in the stocks table for each store, joins with the stores table based on the store ID, groups the results by store ID, store name, and state.

QUERY 15:

```
SELECT DISTINCT order_id, customer_id
FROM ORDERS
WHERE order_id IN (
```

```

SELECT DISTINCT order_id
FROM ORDER_ITEMS
INNER JOIN
    products
ON
    order_items.product_id = products.product_id
AND
    brand_id = 9
AND
    discount >= .20
);

```

```

1  SELECT DISTINCT order_id,customer_id
2  FROM ORDERS
3  WHERE order_id IN (
4      SELECT DISTINCT order_id
5          FROM ORDER_ITEMS
6          INNER JOIN
7              products
8          ON
9              order_items.product_id = products.product_id
10         AND
11             brand_id = 9
12         AND
13             discount >= .20
14 );

```

Output:

Results		Explain	Describe	Saved SQL	History
	ORDER_ID	CUSTOMER_ID			
212		1373			
288		780			
297		1425			
31		1238			
951		339			
955		880			
961		82			
641		673			

Explanation: This query filters records from an unspecified table where the brand ID is 9 and the discount is 20% or higher.

QUERY 16—

```

SELECT
    t1.customer_id,
    MAX(t1.order_date) AS most_recent_order,
    MAX(t2.order_date) AS second_most_recent_order

```

```

FROM ORDERS t1
INNER JOIN ORDERS t2
ON
    t1.customer_id = t2.customer_id
AND
    t1.order_date > t2.order_date
GROUP BY
    t1.customer_id;

```

```

1  SELECT
2  |     t1.customer_id,
3  |     MAX(t1.order_date) AS most_recent_order,
4  |     MAX(t2.order_date) AS second_most_recent_order
5  FROM ORDERS t1
6  INNER JOIN ORDERS t2
7  ON
8  |     t1.customer_id = t2.customer_id
9  AND
10 |     t1.order_date > t2.order_date
11 GROUP BY
12 |     t1.customer_id;
13

```

Output:

Results				Explain	Describe	Saved SQL	History	
	CUSTOMER_ID	MOST_RECENT_ORDER	SECOND_MOST_RECENT_ORDER					
14		04/22/2018	09/04/2016					
72		04/21/2018	09/05/2016					
64		04/04/2018	01/26/2017					
50		04/12/2018	12/07/2016					
27		04/01/2018	12/29/2016					
57		04/20/2018	01/19/2016					
237		04/08/2018	04/11/2016					
51		04/02/2018	05/13/2016					

pg611@scarletmail.rutgers.edu bdmparul en Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 23.2.1

Explanation: This SQL query uses self-joins on the "orders" table to find the most recent and second most recent order dates for each customer, grouping the results by customer ID.

QUERY 17-

```

SELECT DISTINCT order_id, Customer_id
FROM ORDERS
WHERE EXISTS ( SELECT 1 FROM order_items WHERE discount >= 0.20 AND
order_items.order_id = orders.order_id);

```

```

1  SELECT DISTINCT order_id, customer_id
2  FROM ORDERS
3  WHERE
4      EXISTS (
5          SELECT
6              1
7          FROM
8              order_items
9          WHERE
10             discount >= .20
11         AND
12             order_items.order_id = orders.order_id
13     );

```

Output:

Results		Explain	Describe	Saved SQL	History
	ORDER_ID	CUSTOMER_ID			
133		409			
140		1264			
143		693			
146		288			
169		99			
174		1361			
189		1288			
194		558			

pg61@scarletmail.rutgers.edu bdmparul en Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 23.2.1

Explanation: This SQL query selects unique order IDs and customer IDs from the "orders" table for orders where at least one item in the corresponding "order_items" table has a discount of 20% or more, utilizing the EXISTS clause for the subquery condition.

QUERY 18—

```

1  SELECT c.category_id, c.category_name, COUNT(p.product_id) AS total_products
2  FROM categories c
3  LEFT JOIN products p ON c.category_id = p.category_id
4  GROUP BY c.category_id, c.category_name;
5
6

```

Results		Explain	Describe	Saved SQL	History
	CATEGORY_ID	CATEGORY_NAME	TOTAL_PRODUCTS		
2		Comfort Bicycles	30		
7		Road Bikes	60		
1		Children Bicycles	59		
5		Electric Bikes	24		
3		Cruisers Bicycles	78		
6		Mountain Bikes	60		
4		Cyclocross Bicycles	10		

7 rows returned in 0.04 seconds [Download](#)

sb2509@scarletmail.rutgers.edu bdm.shubhamm en Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 23.2.1

Explanation: This SQL query retrieves the category ID, category name, and the count of products for each category, including categories with no products. It uses a LEFT JOIN and GROUP BY clause.

Query 19:

```
1 SELECT SUM(p.list_price * s.quantity) AS total_revenue
2 FROM stocks s
3 JOIN products p ON s.product_id = p.product_id
4 WHERE s.store_id = 1;
5
```

Results Explain Describe Saved SQL History

TOTAL_REVENUE
6487242.16

1 rows returned in 0.01 seconds [Download](#)

Explanation: This SQL query calculates the total revenue by multiplying the list price of products with their corresponding quantities in stock at store_id 1. It involves a JOIN between "stocks" and "products".

Query 20 :

```
1 SELECT product_id, quantity
2 FROM stocks
3 WHERE store_id = 1 AND quantity > 20;
4
```

Results Explain Describe Saved SQL History

PRODUCT_ID	QUANTITY
1	27
4	23
5	22
20	26
21	24
22	29
27	21
30	30

Explanation: This SQL query retrieves the product_id and quantity from the 'sales' table where the store_id is 1 and the quantity is greater than 20. It filters sales data based on specified conditions.

Query21:

The screenshot shows the Oracle APEX interface with a SQL query editor and a results grid. The SQL code is:

```
1 SELECT state, COUNT(customer_id) AS total_customers
2 FROM customers
3 GROUP BY state;
```

The results grid has two columns: STATE and TOTAL_CUSTOMERS. The data is:

STATE	TOTAL_CUSTOMERS
NY	1019
TX	142
CA	284

3 rows returned in 0.03 seconds [Download](#)

Explanation: This SQL query retrieves the count of customers for each unique state from the "customers" table, grouping the results by state. It provides a summary of the total customers in each state.

Query 22:

The screenshot shows the Oracle APEX interface with a SQL query editor and a results grid. The SQL code is:

```
1 SELECT b.brand_id, b.brand_name, COUNT(p.product_id) AS product_count
2 FROM brands b
3 LEFT JOIN products p ON b.brand_id = p.brand_id
4 GROUP BY b.brand_id, b.brand_name
5 ORDER BY product_count DESC
6
7
```

The screenshot shows the Oracle APEX interface with a results grid. The data is:

BRAND_ID	BRAND_NAME	PRODUCT_COUNT
9	Trek	135
1	Electra	118
8	Surly	25
7	Sun Bicycles	23
2	Haro	10
4	Pure Cycles	3
3	Heller	3
6	Strider	3

sb2509@scarletmail.rutgers.edu bdm.shubhamm en Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 23.2.1

Explanation: This SQL query retrieves the brand ID, brand name, and the count of products for each brand, including those with no products. It uses a LEFT JOIN, GROUP BY, and orders results by product count in descending order.

Query 23:

```
1  SELECT
2    |   customer_id,
3    |   COUNT(order_id) AS total_orders
4  FROM orders
5  GROUP BY customer_id;
```

The screenshot shows the Oracle APEX interface with the SQL editor at the top containing the query. Below it is a results grid with two columns: 'CUSTOMER_ID' and 'TOTAL_ORDERS'. The data shows various customer IDs with their corresponding order counts. At the bottom, there are user profile icons and copyright information.

CUSTOMER_ID	TOTAL_ORDERS
763	1
371	1
237	2
459	1
1127	1
1288	1
1102	1
591	1

Explanation: This SQL query selects the customer ID and counts the total orders placed by each customer from the "orders" table. The results are grouped by customer ID, providing an order count per customer.

Query 24:

```
1  SELECT *
2  FROM order_items
3  WHERE discount > 0.15;
```

The screenshot shows the Oracle APEX interface with the SQL editor at the top containing the query. Below it is a results grid with seven columns: ID, ORDER_ID, ITEM_ID, PRODUCT_ID, QUANTITY, LIST_PRICE, and DISCOUNT. The data shows specific items from order_items where the discount is greater than 0.15.

ID	ORDER_ID	ITEM_ID	PRODUCT_ID	QUANTITY	LIST_PRICE	DISCOUNT
503	179	3	18	1	449	.2
504	179	4	12	2	549.99	.2
509	181	3	15	2	529.99	.2
513	183	1	11	2	1680.99	.2
517	184	1	4	1	2899.99	.2
526	189	1	16	2	599.99	.2
528	190	2	15	1	529.99	.2
529	190	3	3	2	999.99	.2

Explanation: This SQL query selects all columns from the "order_items" table where the

discount is greater than 0.15, filtering and retrieving rows with discounts exceeding the specified threshold.

Query 25:

```
1  SELECT *
2  FROM stocks
3  WHERE quantity = 0;
4
```

Results	Explain	Describe	Saved SQL	History
ID	STORE_ID	PRODUCT_ID	QUANTITY	
6	1	6	0	
8	1	8	0	
32	1	32	0	
42	1	42	0	
92	1	92	0	
160	1	160	0	
163	1	163	0	
168	1	168	0	

sb2509@scarletmail.rutgers.edu

bdm.shubhamm

en

Copyright © 1999, 2023, Oracle and/or its affiliates.

Oracle APEX 23.2.1

Explanation: This SQL query selects all columns from the "stocks" table where the quantity is equal to zero, retrieving rows that represent items with no available stock.

Query 26:

```
1  WITH ranked_products AS (
2    SELECT store_id,
3           product_id,
4           quantity,
5           RANK() OVER (PARTITION BY store_id ORDER BY quantity DESC) AS ranking
6    FROM stocks
7  )
8  SELECT *
9  FROM ranked_products
10 WHERE ranking = 1;
11
```

STORE_ID	PRODUCT_ID	QUANTITY	RANKING
1	30	30	1
1	292	30	1
1	219	30	1
1	193	30	1
1	188	30	1
1	109	30	1
1	106	30	1
1	68	30	1

sb2509@scarletmail.rutgers.edu

bdm.shubhamm

en

Copyright © 1999, 2023, Oracle and/or its affiliates.

Oracle APEX 23.2.1

Explanation: This SQL query uses a common table expression (CTE) to rank products by quantity within each store and selects the products with the highest quantity ranking (1) from the CTE.

Query 27:

```
1  SELECT product_id,  
2        MAX(quantity) AS max_quantity,  
3        MIN(quantity) AS min_quantity,  
4        AVG(quantity) AS avg_quantity  
5  FROM stocks  
6 GROUP BY product_id;
```

Explanation: This SQL query retrieves the maximum, minimum, and average quantity values for each unique product_id from the "stores" table, grouping the results by product_id.

Query 28:

```
1 SELECT active, COUNT(*) AS staff_count
2 FROM staffs|
3 GROUP BY active;
4
```

Results	Explain	Describe	Saved SQL	History
		ACTIVE	STAFF_COUNT	
1		10		
1 rows returned in 0.01 seconds		Download		

Explanation: This SQL query counts the number of staff members for each "active" status, grouping results by the "active" column. It provides a count of active and inactive staff members.

Query 29:

```
1  SELECT o.order_id, SUM(oi.quantity * oi.list_price * (1 - oi.discount)) AS total_price
2  FROM order_items o
3  JOIN order_items oi ON o.order_id = oi.order_id
4  GROUP BY o.order_id;
5
6
```

Results	Explain	Describe	Saved SQL	History
ORDER_ID				TOTAL_PRICE
210				1366.1494
226				712.4905
232				31869.6576
237				9861.8288
277				6593.85
281				24531.1623
290				12285.0414
291				2790.3248

sb2509@scarletmail.rutgers.edu

bdm.shubhamm

en

Copyright © 1999, 2025, Oracle and/or its affiliates.

Oracle APEX 23.2.1

Explanation: This SQL query calculates the total price for each order by multiplying the quantity, list price, and discount-adjusted price of items, grouping results by order ID. It involves joining two tables.

Query 30:

```
1  SELECT product_id, SUM(quantity) AS total_quantity_sold
2  FROM stocks
3  WHERE store_id = 1
4  GROUP BY product_id
5  ORDER BY total_quantity_sold DESC
6  LIMIT 10;
7
```

Results	Explain	Describe	Saved SQL	History
PRODUCT_ID				TOTAL_QUANTITY SOLD
64				30
68				30
109				30
30				30
106				30
219				30
292				30
193				30

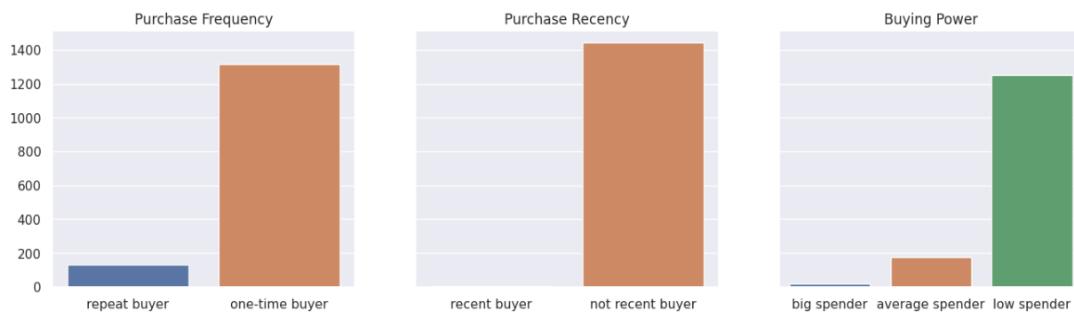
Explanation: This SQL query retrieves the total quantity sold for each product in Store 1, groups results by product, orders them by descending total quantity sold, and limits the output to 10 rows.

Visualization:

Create a line plot showing the 30-day moving average of orders over time, with each store's trend distinguished by color, while removing axis labels and ticks for a cleaner visualization.



Visualize customer characteristics using subplots with count plots for purchase frequency, purchase recency, and buying power, while sharing the y-axis for easier comparison. Remove axis labels and titles for a clean presentation.



Generate a bar plot illustrating the average units sold per month, differentiated by product category, with axis labels removed for clarity, and displaying the error bars.

Average Units Sold by Month of Year

